DTIC FILE COPY

AD-A210 702

1

ARI Research Note 89-14

A Prototype Intelligent Maintenance Tutoring System for Troubleshooting the M16A1 Automatic Rifle

DTIC
S ELECTE D
AUG 0 1 1989
D

Mark L. Miller
Computer* Thought Corporation

for

Contracting Officer's Representative
Merryanna L. Swartz

Technologies for Skill Acquisition and Retention
Technical Area
Zita M. Simutis, Chief

Training Research Laboratory
Jack H. Hiller, Director

March 1989

United States Army
Research Institute for the Behavioral and Social Sciences

Approved for the public release; distribution is unlimited.

89 8 01 049

U.S. ARMY RESEARCH INSTITUTE FOR THE BEHAVIORAL AND SOCIAL SCIENCES

A Field Operating Agency Under the Jurisdiction of the Deputy Chief of Staff for Personnel

EDGAR M. JOHNSON	**JON W. BLADES**
Technical Director	**COL, IN**
	Commanding

Research accomplished under contract for the Department of the Army

Computer* Thought Corporation

Technical review by

Stanley J. Kostyla

Accesion For		
NTIS CRA&I	✓	
DTIC TAB	☐	
Unannounced	☐	
Justification		
By		
Distribution /		
Availability Codes		
Dist	Avail and / or Special	
A-1		

NOTICES

DISTRIBUTION: This report has been cleared for release to the Defense Technical Information Center (DTIC) to comply with regulatory requirements. It has been given no primary distribution other than to DTIC and will be available only through DTIC or the National Technical Informational Service (NTIS).

FINAL DISPOSITION: This report may be destroyed when it is no longer needed. Please do not return it to the U.S. Army Research Institute for the Behavioral and Social Sciences.

NOTE: The views, opinions, and findings in this report are those of the author(s) and should not to be construed as an official Department of the Army position, policy, or decision, unless so designated by other authorized documents.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS --
2a. SECURITY CLASSIFICATION AUTHORITY --	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE --	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) C*T-ARI-880001	5. MONITORING ORGANIZATION REPORT NUMBER(S) ARI Research Note 89-14

6a. NAME OF PERFORMING ORGANIZATION Computer*Thought Corporation	6b. OFFICE SYMBOL *(If applicable)* --	7a. NAME OF MONITORING ORGANIZATION U.S. Army Research Institute for the Behavioral and Social Sciences
6c. ADDRESS *(City, State, and ZIP Code)* 840 Avenue F, Suite 104 Plano, TX 75074-5000		7b. ADDRESS *(City, State, and ZIP Code)* 5001 Eisenhower Avenue Alexandria, VA 22333-5600
8a. NAME OF FUNDING/SPONSORING ORGANIZATION U.S. Army Research Institute for the Behavioral and Social Sciences	8b. OFFICE SYMBOL *(If applicable)* PERI-IC	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER MDA903-87-C-0573
8c. ADDRESS *(City, State, and ZIP Code)* 5001 Eisenhower Avenue Alexandria, VA 22333-5600		10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
63007A	795	336	C2

11. TITLE *(Include Security Classification)*
A Prototype Intelligent Tutoring System for Troubleshooting the M16A1 Automatic Rifle

12. PERSONAL AUTHOR(S)
Miller, Mark L.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Year, Month, Day)*	15. PAGE COUNT
Final	FROM 88/04 TO 88/12	1989, March	44

16. SUPPLEMENTARY NOTATION
This report presents the results of a Phase I SBIR award to Computer*Thought Corporation, executed in cooperation with Rediffusion Simulation, Incorporated.

17. COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB-GROUP	Intelligent tutoring systems ITS EIDS [illegible] ← Courseware
			Artificial intelligence ICAI Training
			Expert systems CBT Student modelling

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*
This report describes a research and development effort to investigate the feasibility of combining two new instructional technologies--intelligent tutoring systems and interactive videodisc-based courseware--into a single instructional vehicle operable on low-cost personal computers. Funded by the Army Research Institute through the Small Business Innovative Research program, the project developed a demonstrable prototype that combines a rule-based expert system with a videodisc-based instruction delivery package for troubleshooting M16A1 automatic rifles.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Merryanna L. Swartz	22b. TELEPHONE *(Include Area Code)* (202) 274-5789	22c. OFFICE SYMBOL PERI-IC

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

ACKNOWLEDGMENTS

The author of a report such as this receives undue credit for the ideas and efforts of others. I am truly indebted to numerous people and several organizations for the hard work leading up to this report.

At Computer * Thought Corporation (C*T), key contributors included A. Ramsay Ellis, Jr., and Janet L. Miller. Crucial contributions were made by our partners at Rediffusion Simulation, Incorporated: Tom D. Conkright, Ph.D., Scott R. Lucado, and Michael J. Wild. Scott Lucado and Ramsay Ellis, in particular, provided an extraordinary team effort in meeting the substantial technical challenges of the project. At C*T, we would also like to thank Arthur James and Alice Marie Miller for their assistance; and, at RSI, Karen Pogoloff and Bob Stevenson, who have provided valuable suggestions and moral support every step of the way. Dr. Dennis Irons of Texas Instruments contributed insights regarding tool selection and instructional systems design.

We are grateful to our subject matter expert, Master Sergeant Tom Coppock, for sharing his insights and anecdotes on M16 riflery training. We also wish to thank Staff Sergeant Eric Weaver, Technical Sergeant Allen West, the 136th Combat Arms Training and Maintenance Unit, the Texas Air National Guard, and Carswell Air Force Base for allowing us to participate in and videotape their training sessions.

Special thanks are also due to Doug Lising and Tom Wooten at Fort Eustis for helping us to obtain three key videodiscs; and to Stan Kostyla, Joe Psotka, and Merryanna L. Swartz, among others, at the U.S. Army Research Institute for the Behavioral and Social Sciences (ARI), for helpful editorial suggestions, for keeping us on track, and for having the vision to encourage and fund this investigation.

A PROTOTYPE INTELLIGENT MAINTENANCE TUTORING SYSTEM FOR TROUBLESHOOTING THE M16A1 AUTOMATIC RIFLE

CONTENTS

A PROTOTYPE INTELLIGENT MAINTENANCE TUTORING SYSTEM FOR TROUBLESHOOTING THE M16A1 AUTOMATIC RIFLE

INTRODUCTION

The Army Problem

During recent decades, the U.S. Army -- indeed, every branch of the the U.S. armed services -- has become increasingly asset-intensive (as opposed to *personnel*-intensive). Equipment and weapons systems have become more elaborate and more expensive, requiring greater skills to operate, maintain, troubleshoot, and repair. By contrast, the Army has found it increasingly difficult to attract and retain qualified, adequately trained, technical personnel. Many new recruits require remediation in such basic skills as reading and algebra.

The impact of this widening gap between personnel skills and technology on operational readiness is becoming apparent. Properly functioning modules are returned to depots for repair. Depots send improperly functioning units back to the field for deployment. More and more, the Army must depend on civilian experts and instructors, who command higher pay scales, yet tend to be unfamiliar with military considerations and procedures.

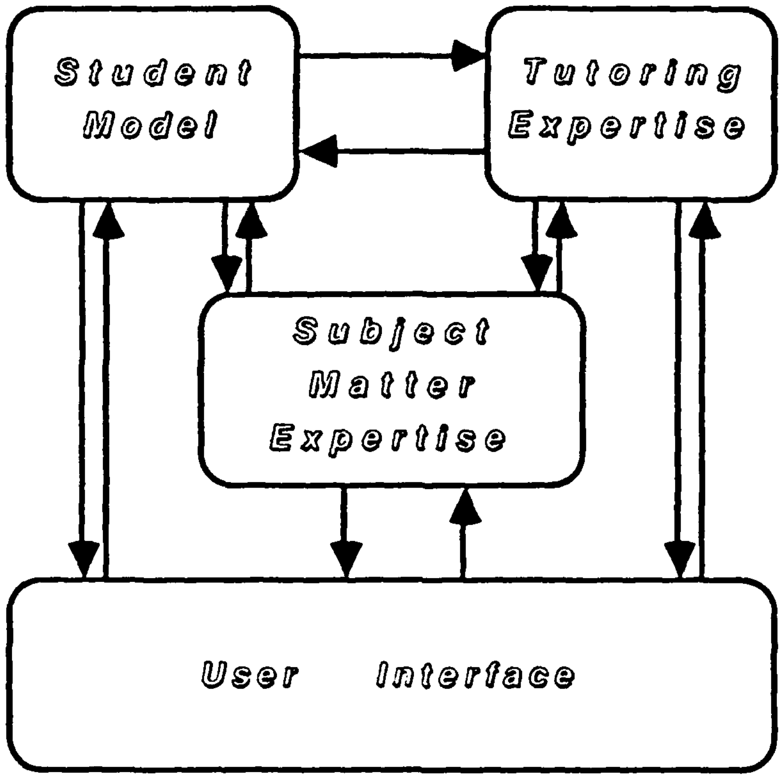
Conversely, the availability of quality technical training appears to be a key factor in the Army's ability to recruit high calibre individuals. These and other considerations argue for the application of the most advanced instructional technologies, provided that these approaches can be delivered cost effectively, in volume, to meet the Army's technical training requirements.

Potential Impact of Intelligent Tutoring Systems

One approach that has captured the imagination of researchers and educators has been termed, *intelligent tutoring systems* [ITS] or *intelligent computer-aided instruction* [ICAI]. The goal of ITS research is to provide a new plateau of instructional capability by integrating artificial intelligence [AI] and expert systems [ES] techniques, including several types of knowledge, into next-generation training systems. This goal includes task knowledge, gleaned from human subject matter experts as well as insight into student reasoning, skill acquisition, and tutorial strategies, gleaned from skilled human instructors. Figure 1 illustrates the major modules present in a typical ITS architecture (Woolf, 1984).

A series of attempts, from SCHOLAR (Carbonell, 1970), through SOPHIE (Brown *et al.*, 1982), to current projects funded by ARI, such as Bolt, Beranek, and Newman's HAWK Radar Tutor (De Bruin *et al.*, 1988) have moved us ever closer to that goal. However, ITS research, to date, has lacked the impetus that ES work has already gained from a few landmark success stories, such as DEC's famous XCON system (McDermott, 1982). Moreover, the more impressive prototype systems constructed to date have operated on large timesharing computers (such as the DECSYSTEM 2060), or dedicated, LISP-based workstations (such as the SYMBOLICS 3670). For ITS technology to achieve its full potential, a more practical approach leading to an incontrovertibly successfully application is needed.

Figure 1: The Major Modules in an Intelligent Tutoring System



Potential Impact of Interactive Videodiscs

Even on the most expensive equipment, ITS researchers have tended to settle for minimal graphics with less-than-convincing realism. Many ITS systems are entirely text-based, an unacceptable user interface for many applications where the target population lacks skill in reading comprehension. By contrast, impressive authoring environments have been developed using interactive videodisc [IVD] technology to provide motion video sequences, realistic still-frames, audio output including digitized speech, and graphics-over-video capability. The Army has recently adopted a new standard in applying this exciting technology to its training and dissemination requirements: the electronic information delivery system [EIDS], manufactured by Matrox Electronic Systems LTD (Quebec, Canada).

Unfortunately, all too often, the interactive videodisc -- with its potential for massive data storage and random access retrieval -- is used much like a videotape player. Many computer-based training [CBT] packages ("courseware") incorporate lengthy sequences of linear video, followed by a lock-step multiple choice quiz format, leaving students bored and frustrated. All but the most sophisticated authoring tools tend to force developers into a rigid, frame-based, exhaustive branching structure for courseware, the very anti-thesis of the ITS goal. What is needed, then, is to integrate the best of ITS, CBT, and IVD technologies, and to reduce the result to cost-effective practice.

Target Population

The target population for the training system was selected to be broad-based and typical of the Army's overall technical training requirements. Every new Army recruit undergoes M16 Riflery Qualification, regardless of later specialization. However, since current-generation ES technology lends itself better to applications such as diagnosis (rather than, say, marksmanship), only the troubleshooting aspect of M16 riflery was investigated. In particular, the target knowledge base was chosen taking into account U.S. Army policy regarding the extent of field take-down and repair. The lower receiver of the M16A1, for example, is made of aluminum to keep the weapon lightweight, but the pins holding it together are made of steel, which results in wear-and-tear as a result of excessive assembly and disassembly. Hence, field personnel are not authorized to disassemble the lower receiver at all. The expert system models the knowledge and doctrine applicable to field personnel; it does not include the expertise required of an armorer or of depot personnel. Hence, it contains no knowledge of the inner parts or operation of the lower receiver. This constraint was helpful in bounding the scope of knowledge required for the expert module, as well as clarifying the training objectives of the overall system. One way to get an appreciation of the training problem for the intended audience is to read the "Sally" manual (Army, 1969) used in current training; Figure 2 shows a typical page from this publication.

Target Hardware

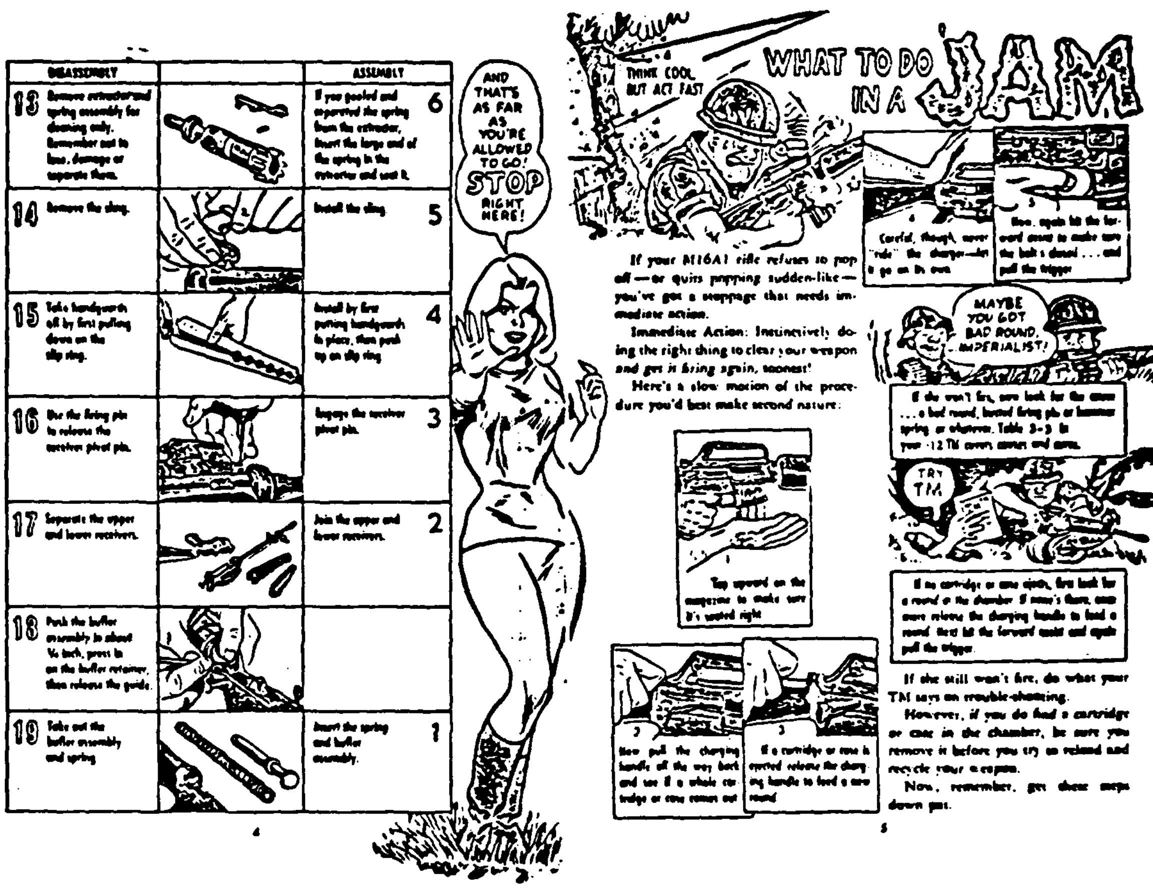
A primary object of the investigation was to determine the feasibility of delivering ITS technology on widely available, low-cost hardware. Had EIDS equipment been available in time for Phase I, this equipment would have been selected as the ideal candidate for such a feasibility investigation. However, since Matrox was not yet geared up for full-scale production of EIDS units, the most similar possible existing equipment was chosen instead. A PC/AT-compatible computer based on an Intel 80286 processor, with up to 640K bytes[1] of random access memory [RAM], rigid disk, and an IBM-compatible enhanced graphics adapter and monitor was selected; this was augmented by an optional ramdisk memory extension card, an ONLINE graphics-overlay/video-controller board, a Sony LDP 2000 videodisc player, and a Microsoft 2-button mouse.[2] To convert the resulting prototype to EIDS involves developing a TENCORE device driver for EIDS similar to that used for the ONLINE board, and reducing RAM utilization to 512K bytes.

A more convenient approach might have been to use a more powerful computer, such as one based on the Motorola 68020 or Intel 80386 micro-processors, allowing for up to 24

[1]It has subsequently been determined that the standard Army configuration for EIDS supports only 512K bytes. (Commercial EIDS configurations support a memory extension board.) Further reductions of memory utilization, to accommodate the more limited Army configuration, are planned.

[2]The software has been shown to operate on IBM equipment, Zenith equipment, and other compatibles running DOS 3.2, also using a Sony LDP 1000 and a Mouse systems compatible 3-button mouse.

Figure 2: Typical Page from "Sally" Manual for M16A1 Rifle


DISASSEMBLY
ASSEMBLY
AND THAT'S AS FAR AS YOU'RE ALLOWED TO GO! STOP RIGHT HERE!
THINK COOL BUT ACT FAST
WHAT TO DO IN A JAM
If your M16A1 rifle refuses to pop off—or quits popping sudden-like—you've got a stoppage that needs immediate action.
Immediate Action: Instinctively doing the right thing to clear your weapon and get it firing again, soonest!
Here's a slow motion of the procedure you'd best make second nature:
MAYBE YOU GOT BAD ROUND, IMPERIALIST!
TRY TM
If she still won't fire, do what your TM says on trouble-shooting.
Now, remember, get these steps down pat.

megabytes of RAM. Another, also easier, solution might have been to use two PC/AT's, one for TENCORE+ and another for OPS5+, each dedicated to its own application. However, such schemes neglect the economic need for cost-effective, high volume delivery, as well as the unprecedented potential for training impact offered by the Army's adoption of the EIDS standard.

Relation to Prior Work

Although a thorough literature review is beyond the scope of this report, readers with varying backgrounds may desire a starting point for further readings. The project has been influenced by two primary threads. The ITS background is currently best summarized in a collection by Sleeman and Brown (1975); more recent work is reported in a collection by Psotka *et al.* (1988). The systems approach to training [SAT], supported by the TENCORE+ authoring environment, grew out of early work on systems such as PLATO and TICCIT (Alessi and Trollip, 1985).[3] Fletcher (1985) has been a consistent proponent for the potential impact of both intelligent tutoring systems and interactive videodisc technology on military technical training.

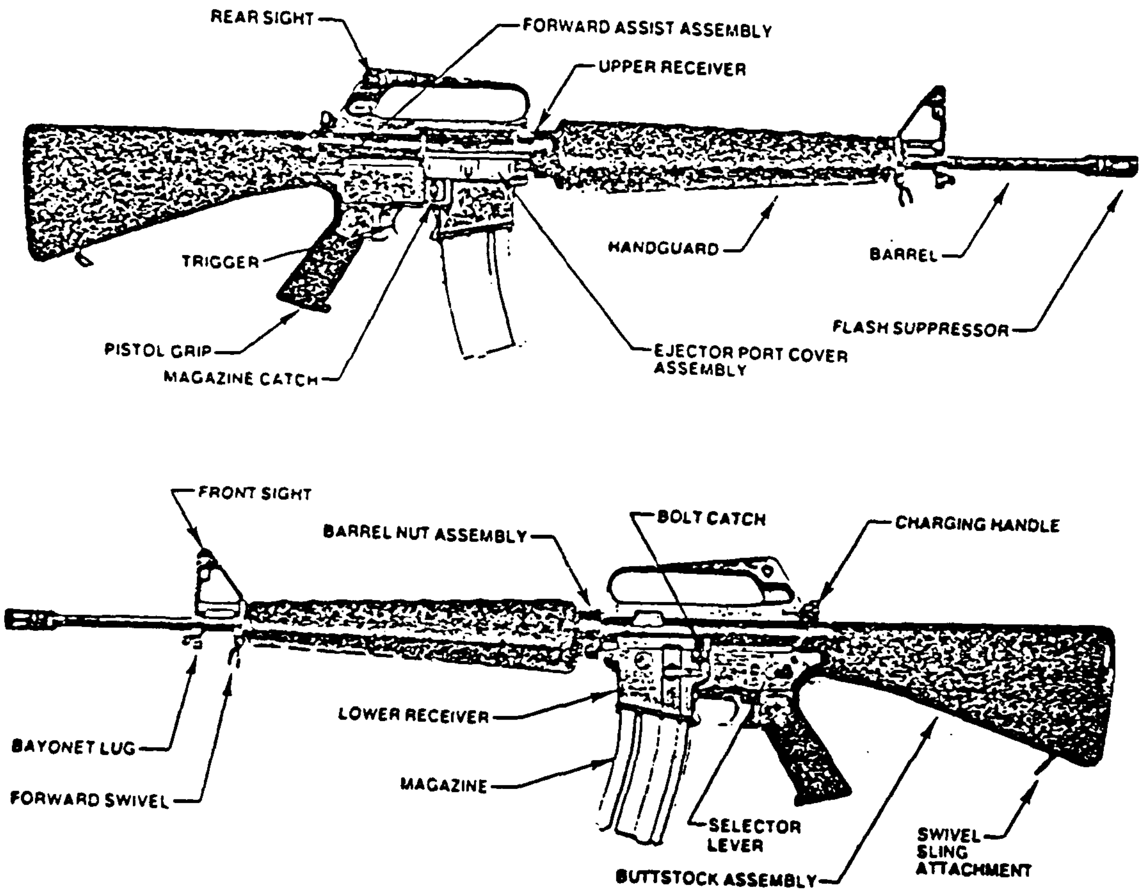
[3]The terminology, *instructional systems design [ISD]*, may be more familiar to some readers. The U.S. Army has recently begun recommending use of the alternative phrase, *systems approach to training [SAT]*, which has been adopted throughout this report.

METHOD

Selection of M16A1 Rifle as Example Topic

The M16A1 rifle was selected an an example device for Phase I of this investigation for several reasons. First, training soldiers to troubleshoot this widely-used small arm has direct relevance to Army needs. Also, the device is simple enough to be thoroughly analyzed, yet complex enough to require non-trivial knowledge and reasoning. This is illustrated by the numbers and relationships of the parts in Figures 3, 4, and 5.

Figure 3: The M16A1 Automatic Rifle



Finally, the U.S. Army was able to provide existirg videodiscs about the M16 rifle family, allowing the prototype system to incorporate both live video sequences and realistic still-frames, without developing new video materials.

Figure 4: The Major Sub-Assemblies of the M16A1 Rifle

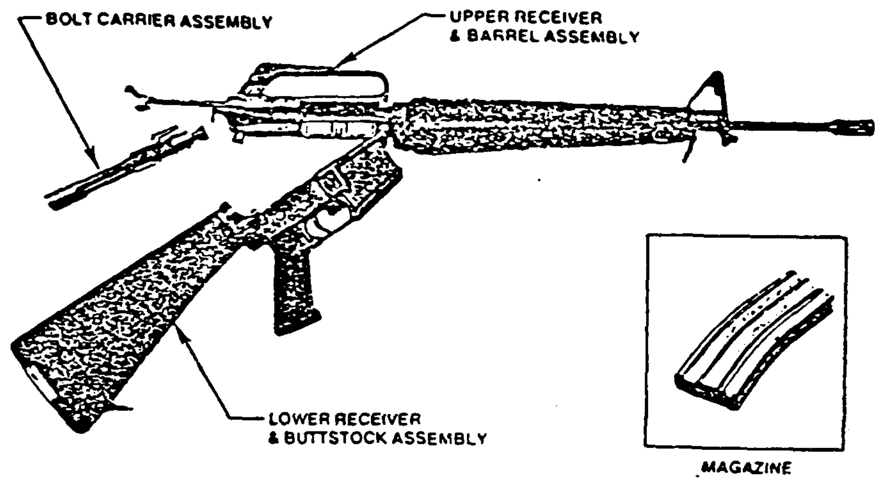
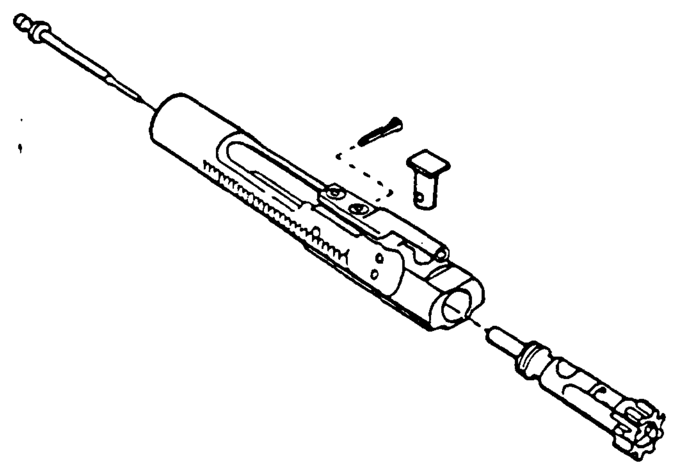

BOLT CARRIER ASSEMBLY
UPPER RECEIVER & BARREL ASSEMBLY
LOWER RECEIVER & BUTTSTOCK ASSEMBLY
MAGAZINE

Figure 5: The Bolt Carrier Sub-Assembly of the M16A1 Rifle



Selection of OPS5+ as Expert System Tool

OPS5+ is an expert system development tool produced by Computer * Thought Corporation. C*T's competitive interest in using its own tool, however, was not the major factor in its selection. OPS5+ is *C*-based, rather than LISP-based, resulting in relatively high performance even on a minimal PC configuration. Its pseudo-code representation of production rules offers a careful balance between execution speed and memory usage; purely compiled approaches tend to require additional memory. A key consideration was its flexibility, including the ability to have rules call on *C*-based modules, for interfacing with a CBT authoring system. Another important factor was C*T's ownership of the source code, familiarity with internal operation, and ability to modify the product. Although the intent was to build an interface between unmodified, off-the-shelf modules, having the source code provided an important hedge against unforeseen technical difficulties. Finally, it was thought that a straightforward rule-based paradigm -- as opposed to a more elaborate hybrid package incorporating rules, frames, and other representational schemes -- would be adequate for capturing the subject matter knowledge and more readily lend itself to student modelling.

Selection of TENCORE+ as CBT Authoring Environment

The TENCORE authoring language is similar to the TUTOR language on PLATO, but it has been adapted for PC delivery and undergone dramatic enhancements in graphics device support and overall functionality. It was selected for use in this study for several reasons. One factor was Rediffusion Simulation's prior experience in using the tool, including existing tool packages for pull-down menus and other capabilities. The primary factor was its high ranking in a comparative evaluation performed at the beginning of this project, with respect to flexibility and functional specifications. Unlike many PC-compatible authoring systems, TENCORE+ does not lock the courseware developer into a specific format or organization. In particular, the ability to write data files readable by the expert system, PAUSE to DOS, and execute an arbitrary DOS program were crucial. Another factor was the potential for moving to an EIDS delivery environment in Phase II, although EIDS equipment was not yet available in time for use during Phase I.

Knowledge Acquisition and Task Analysis

Initial project efforts focused on the acquisition of M16 riflery knowledge through a variety of sources. Reading materials included both informal field manuals aimed at infantry recruits (Army, 1969), and formal technical manuals aimed at depot personnel, including (Army, 1983) and (Colt, 1980). This was followed up by formal classroom instruction, including hands-on disassembly of the rifle and marksmanship practice on the range. One of the classroom instructors also served as our primary subject matter consultant, leading smaller-group sessions i- which a variety of questions were answered and hypothetical cases analyzed. Much of this -ining time was captured on VHS-format videotape for later review and analysis. All project -rsonnel had either had prior small arms training (such as with the very similar AR15 rifle) or attended the formal M16A1 Qualification course.

Project personnel with predominantly AI/ES backgrounds (primarily from C*T) then examined the materials from a knowledge engineering perspective, asking questions such as:

- what knowledge is required to troubleshoot rifle malfunctions?
- how should this type of knowledge be represented?

Project personnel with predominantly SAT backgrounds (primarily from RSI) examined the same materials from a task analysis perspective, asking questions such as:

- what are the instructional objectives of the courseware?
- what partial or simplified tasks can be taught earlier in the sequence?

Knowledge Representation

Selection of a particular tool tends to impose a particular epistemology, or at least establish certain boundary conditions for the world-view of the expert module. The OPS5 language divides knowledge into working memory elements [WMEs] and production rules. Its inference engine operates in a strictly forward chaining fashion, without certainty factors or built-in explanatory capabilities. Its lean philosophy gains performance and flexibility over most backward chaining tools. In OPS5+, in particular, a problem space can be searched in a variety of ways, using meta-rules to control even the relative priority of other rules. OPS5 notation, in appearance, is more like LISP code than, say, the more English-like format of tools based on EMYCIN.[4] The flexibility of OPS5+ improves the ability of the resulting application to perform diagnosis in a manner similar to the SME. It also facilitates modelling the reasoning processes of trainees. Sample OPS5+ code from the M16A1 diagnostic ES is provided in Appendix C.

Diagnostic applications tend to lend themselves to backward chaining. Fortunately, by using rules which establish goals and subgoals, backward chaining can be performed conveniently within the framework of a forward chaining engine. The M16A1 expert module actually uses a hybrid search strategy, which was readily derived from the guidelines in technical manuals and examination of scenarios with the SME. Table 1 illustrates the typical format of diagnostic information in military technical manuals.

A good portion of this type of knowledge can be encoded in the form of tables of WMEs, rather than rules, taking advantage of what would otherwise be highly redundant structure across groups of rules for symptom clusters.

The subject matter expertise being modelled is only one of the types of knowledge required in an ITS. Other categories of rules include:

- ***bookkeeping rules*** (for updating rifle state or displaying trace information on a second screen for the benefit of the knowledge engineer);
- ***communication rules*** (for controlling interactions with TENCORE+, such as setting up a hint to be given by TENCORE+ upon student request);

[4]A thorough review of commercially available ES development tools is beyond the scope of this report. For an introduction to the essential concepts and better known tools such as EMYCIN, see (Waterman, 1986).

Table 1: Typical Diagnostic Information in Technical Manuals

Malfunction	*Test or Inspection*	*Corrective Action*
Failure to Fire	Broken Hammer	Replace Hammer
Failure to Fire	Weak Hammer Spring	Replace Spring
Failure to Fire	Broken Firing Pin	Replace Firing Pin
Failure to Cock	Worn Trigger Nose	Replace Trigger
Failure to Cock	Worn Hammer Trigger Notch	Replace Hammer
Failure to Cock	Missing Disconnector Spring	Replace Spring
Failure to Extract	Badly Pitted Chamber	Replace Barrel Assembly

- ***student modelling rules*** (for keeping student-specific parameters, session history and records, and, eventually, modelling the student's knowledge state and learning process);
- ***tutorial strategy rules*** (for encoding tutorial knowledge such as when to interrupt the normal courseware flow due to a serious student error, and, eventually, how to adapt the content and style of feedback based on instructional history or even the learner's cognitive style).

The OPS5 system architecture does not impose a clear division of this knowledge into the isolated modules characteristic of a traditional data flow diagram. Interactions between different groups of rules are prevented by using such production rule programming devices as mode slots in the left-hand-side patterns of rules.

Systems Approach to Training

The process of developing instructional systems using a courseware authoring system involves four steps: analysis; design; development and implementation; and validation/ evaluation. The final step, validation/evaluation, will not discussed here, since it would have been premature, given the exploratory nature of the Phase I program.

Analysis. The analysis step, discussed earlier, parallels the knowledge acquisition step for the ES. The knowledge engineer may be slightly less concerned with behavioral analysis; but the synthesis task requires greater rigor. Cutting through the technical jargon of two seemingly diverse disciplines, however, the underlying processes are far more striking in their similarities than in their differences.

Shared by the practitioners of both AI/ES and SAT is an example-driven or case study methodology. To understand a portion of a task or a chunk of knowledge, it is necessary to see it in action in a specific context. Hence, a set of troubleshooting scenarios were developed to guide further work. The ES would pass acceptance if it could solve all of these scenarios using any approach that a student might find; the courseware would pass acceptance if it could set up each scenario using realistic video sequences and accept the student actions to work through each scenario showing the modified rifle state at each step.

Table 2 shows the four scenarios that were included in the final demonstration system. (Two additional scenarios were designed and partially implemented, but pruned from the demonstration due to memory limitations and schedule constraints.)

Table 2: M16A1 Troubleshooting Scenarios

	Symptom	*Context*	*Solution*
1.	Failure to Fire	Practice Range	Safety On
2.	Failure to Feed	Combat	Improperly Seated Magazine
3.	Fires With Safety On	Cleaning	*Return to Depot*
4.	Failure to Extract	Practice Range	Carbon Buildup

Design. Because of the nature of the project and the need to interface with the ES module, the SAT process was not entirely conventional. The need for an initial data gathering session was recognized, and the eventual need for traditional tutorial presentations, to provide background knowledge about the rifle, was recognized. However, it was decided to develop only stub versions of these, in order to focus available resources on the role of the ES in instruction. The benefits of the ES are most clear in problem solving sessions, such as working through the scenarios mentioned above.

One key idea was the representation of student problem solving steps as verb-object pairs. Every action taken by the student during the solution of a troubleshooting scenario is modelled as the selection of a rifle part and the selection of an action or test operation on that part. The student's complete problem solving protocol can be concisely expressed in this format. Upon recognizing this aspect, the ES module was re-worked so that it used precisely the same set of rifle parts and actions in its own solutions. Thus, a student solution and an expert solution to any scenario could be directly compared.

Another key idea solved the problem of excessive keyboard input and difficult terminology. An M16 riflery trainer that required the student to enter, for instance, "check the extractor lips for carbon buildup" would not be useful for training soldiers. Hence, a pointing device interface was specified for all interaction except initial entry of routine user information such as name, rank, and serial number. A variety of pointing devices are supported by TENCORE+. A mouse was selected for this project because it lends itself to pull-down menus (similar to Apple's Macintosh®). During a scenario, all user interaction is through pointing at mouse-sensitive screen regions, such as to select a rifle subassembly, or pull-down menus, such as to select a particular action to perform on that part.

Appendix A shows sample storyboards used in the design of the TENCORE+ portions of the system.

Development and Implementation. Since an important objective of this research was to demonstrate feasibility using off-the-shelf components, existing Army-supplied videodiscs were used.[5] The use of existing video saved considerable project resources, but restricted the courseware authors to a process of examining the available frames and selecting the most appropriate. To some extent, the availability of suitable video and/or audio accompaniment affected the definition of scenarios. An attempt was made to emphasize scenarios for which suitable motion sequences or crisp still frames could be found. In order to demonstrate a full range of capabilities, scenarios using the video channel for motion, for still-frame, for graphics-only, and for graphics-over-video were chosen; additional efforts were made to ensure that the audio channel was used for sound effects, for music, and for speech.

From a pedagogical standpoint, allowing the availability of these "special effects" to influence the curriculum would seem shocking. However, the objectives of the Phase I effort were to demonstrate a range of capabilities rather than to produce an instructionally sound courseware module. It is crucial that this point be explained as an integral part of system demonstrations. A good example of this weakness involves the use of video sequences to illustrate the results of poor choices by the student. It is plausible that a video motion sequence, especially with audio accompaniment, serves as a positive reinforcer; yet, the current demonstration might be almost as apt to provide such reinforcement for incorrect responses as it is for correct ones. This reflects neither an inherent limitation of the technology nor a lack of appreciation of such issues by the development team; rather, it represents a fruitful topic for further investigation during Phase II.

Design Principles

Several principles guided the design of the system to meet project objectives within the available resources. These were as follows.

- Use off-the-shelf components whenever possible;
- Most decision-making and answer-analysis should be handled by OPS5+;
- All user interfaces should be handled by TENCORE+;
- Each of OPS5+ and TENCORE+ should be restricted to its own screen (the OPS5+ screen is for demonstrating the reasoning trace, only, and would not be seen by actual students);
- Although DOS requires one package to be the parent, the conceptual interface should be more like cooperating co-routines;
- Each of OPS5+ and TENCORE+ should be able to re-start after interruption by examining a state vector specifying which scenario, the state of (dis)assembly of the rifle, the previous steps taken by the student, and similar information;
- The expert system should be able to solve each scenario from any legal state that could arise as a result of student exploration; it should be capable of being reset after each student step;

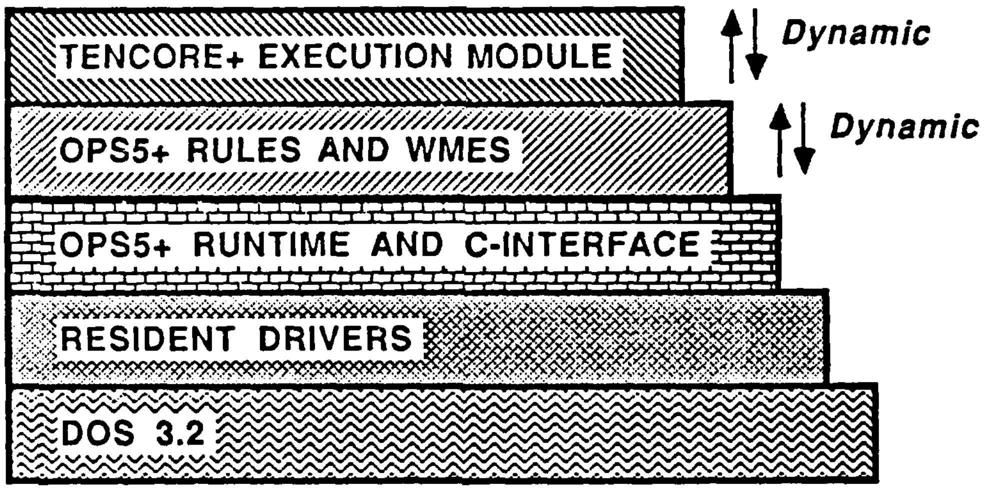
[5]Since three videodiscs were supplied, the relevant frames were re-mastered onto a single new disc, by RSI; however no new video materials were produced during the project.

- The interface should be simple and relatively hardware independent; hence, eg., data should be passed in a file rather than, say, using a reserved block of memory on the ONLINE board;
- The interface design should be as generic as possible and not tied to the rifle troubleshooting task; although some data structures might require adaptation for another topic, the basic structure should not.

System Architecture and Key Interfaces

The mechanics of the system architecture were largely imposed by the DOS operating system and its 640K byte memory limitation. The arrangement of memory is as shown in Figure 6.

Figure 6: Memory Organization in Prototype Tutoring System



OPS5+ is the driver. At each step, it creates a new copy of the TENCORE+ student executor, which initially suffers from a problem we called, "non-resident amnesia." To proceed, after almost every mouse-click, TENCORE+ must read a state file and determine what is on the screen and the student's current options. Surprisingly, speed (such as for screen refresh when switching between modules) was not a problem. The original design called for keeping the key files in ramdisk; however, for demonstrations it was necessary to insert delays so that viewers could even detect module switching.

ITS System Integration

The long-range goal of the research is to deliver a fully functional ITS integrated with an interactive videodisc-based instructional system on low-cost Army-standard equipment. This implies a commitment to provide student modelling and sophisticated tutorial strategies, as well as a representation of subject matter expertise. The "hooks" to provide these additional capabilities have been designed into the system.

Since the primary objective of the Phase I SBIR was to demonstrate the feasibility of cost-effectively integrating ITS and IVD technologies, implementation of a full-scale ITS system was not attempted. However, the availability of an operational expert system for the task offered the opportunity to provide a level of interaction not possible in a conventional CBT framework. Key issues included exploiting the availability of the ES without full student modelling. The expert module could be considered "semi-articulate" since it contained some explanatory capability regarding its own actions. The system supports *learning by imitating* by giving a *specific hint* (what *its* next step would be in solving the scenario), and *learning by guided doing* by giving a *general hint* (what generalized question or hypothesis *its* next step would be trying to address). It also supports various degrees of positive reinforcement, after every student step, not based on canned branching, but based on the actual rule firings of the ES.

By resetting the state of the system after each student step, the expert module is forced to track the *student's* solution trajectory. It is far more useful, from the student's perspective, to see how to finish the current solution -- or to see why it is a blind alley -- than it is to be told about some other approach selected by the expert. Supporting this sort of interaction -- helping the student solve it his/her own way -- would only be possible in a system containing operational expert knowledge. Likewise, the system can tolerate legal but questionable moves by the student, such as side excursions that add steps to a solution but do not prevent completion of the exercise. This would be an area for a more complete tutorial module to determine when to intervene. Currently, wrong moves are classified as non-fatal or fatal; on fatal moves, the student's solution is cut short. Since this training is intended to address real-world situations, possibly including actual combat, in some scenarios it *is* possible for the student to be "killed" for a wrong move. Thus, this realization of a wrong move in a situated training scenario can be highly instructional in military training.

RESULTS AND DISCUSSION

Approximately 200 expert rules were encoded using Computer * Thought Corporation's OPS5+ tool, operating on a PC/AT-compatible personal computer. This rule set embodies sufficient knowledge of M16A1 troubleshooting to solve a variety of scenarios including a range of presenting symptoms (*eg*., failure to eject a spent cartridge), taking into account multiple contexts (*eg*., combat versus firing range), and allowing for the possibility of multiple, interacting underlying causes (*eg*., bent extractor and carbon buildup in chamber). To conserve memory, a subset of the full expert system sufficient to solve the actual training scenarios has been incorporated into the demonstration software.

The expert system was interfaced to the student execution module of the TENCORE+ authoring system for computer-based training. Key design principles included allocating most decision-making and answer analysis processes to the expert system, whereas all user interface operations (such as screen input-output) were left to the authoring system. A set of mouse-sensitive screens illustrates the M16A1 in various states of disassembly, allowing students to interact with the system without typing or recalling the precise terminology for each rifle part. A set of pull-down menus allows the student to select various operations such as clearing the rifle, removing a sub-assembly, or cleaning a part. The set of rifle parts and the available operations on those parts, as seen by students, are identical to those available to the expert system.

Sample Sessions

Figure 7 shows a flow diagram for a typical session with the prototype.

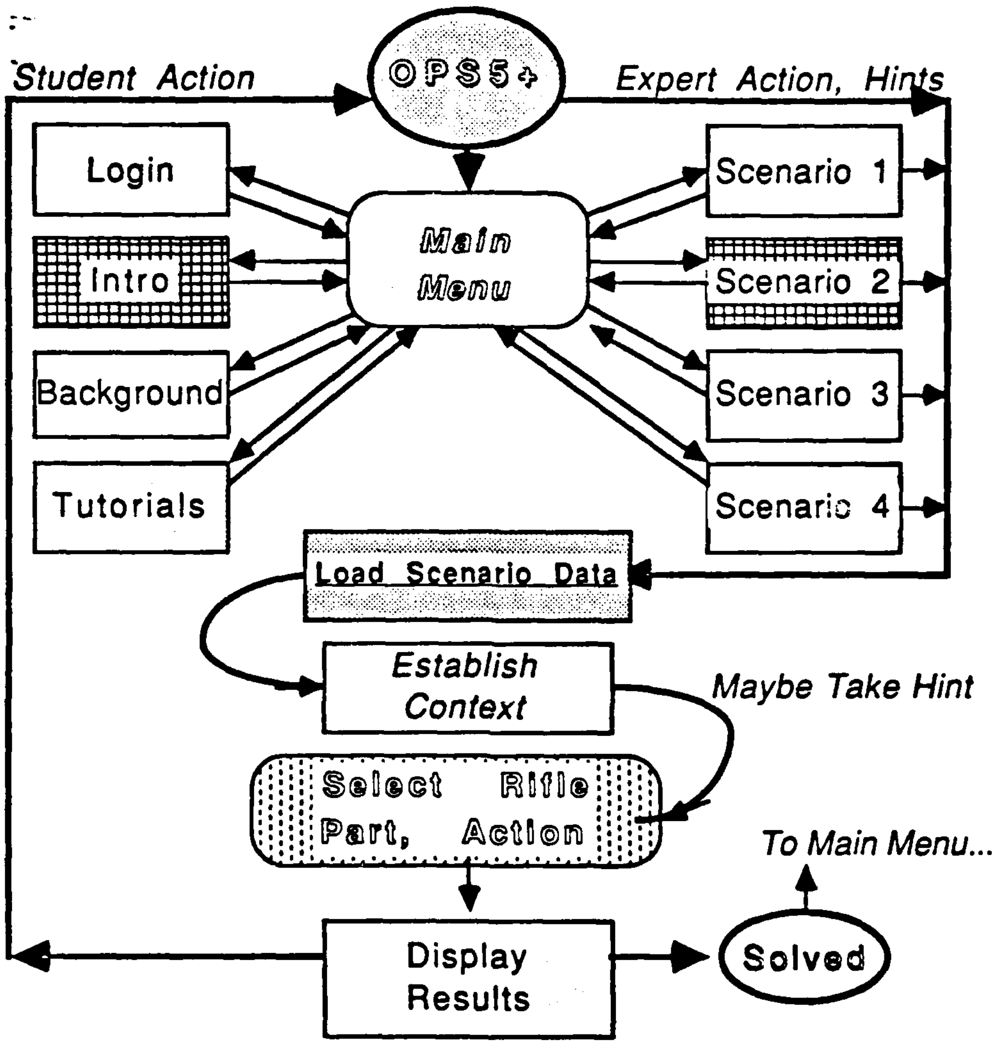
Figures 8 through 9 illustrate the sequence of screens for a single session from a student's login to leaving the system. Scenario 4 has been used since it involves the most detailed reasoning.

Notice the operation of the ES on the second screen. Not every hint that is set up has been taken by this student, of course, but notice how the hints are generated based on the expert solution.

Figures 10 through 12 highlight certain screens from the other three scenarios.[6]

[6]A VHS-format videotape (0.5 inch cassette), illustrating the operation of the system on several scenarios, has been prepared as a supplement to this report. Interested readers should contact the author directly for information about this videotape.

Figure 7: Flow of Control and Key Data During Typical Session

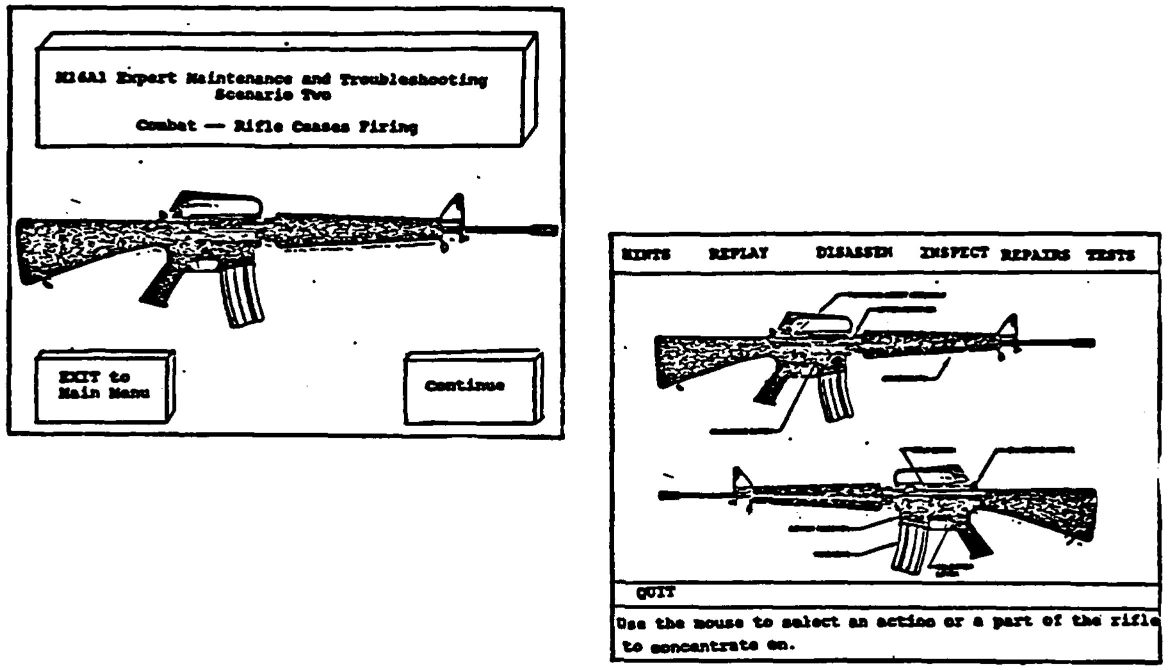


Findings

The primary finding of this study is that a carefully-engineered intelligent tutoring system can be integrated with an interactive videodisc-based courseware package and delivered on low-cost, Army-standard, PC-class hardware.

A key feature of the prototype is the ability of the expert system to single-step its solution, resetting its current state at each step to match the actual step taken by the student. This allows the expert system to closely track student solutions, even when multiple, acceptable solution trajectories exist, or when the student has pursued a relatively harmless blind alley. At each step, the system is able to provide feedback and hints, based not on canned branching within the courseware, but generated on-the-fly, based on the rules actually firing within the expert system.

Figure 8: Sequence of Screens for Sample Session, Part 1



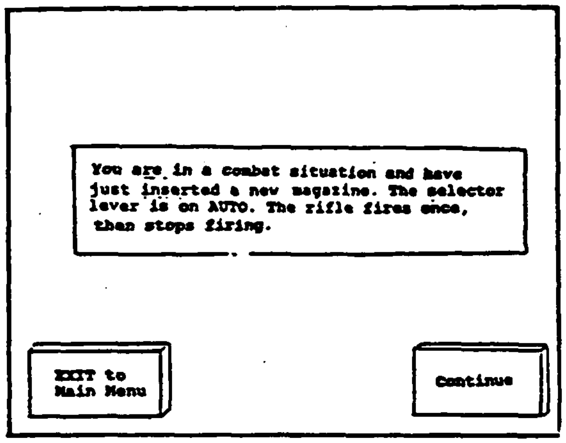
Use the mouse to select an action or a part of the rifle to concentrate on.

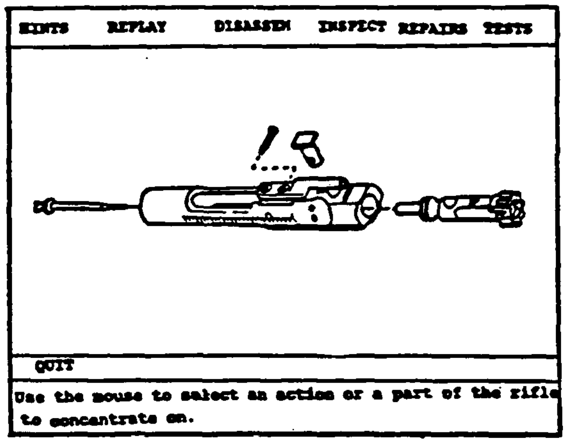
Provisions for student modelling and sophisticated tutorial strategies were also designed, but were not implemented in the Phase I prototype. The system has been demonstrated to several agencies within the military training community. Reactions to the capabilities exhibited by the demonstration system have been enthusiastic.

Additional findings include the observation that processor speed was not the major obstacle to delivery on the PC/AT. However, the 640K memory limitation caused considerable additional effort. It was in fact necessary to re-compile the OPS5+ package with source code modifications to reduce memory utilization. Some interactions between OPS5+ and TENCORE+ could only be resolved by such modifications, as well; so the *off-the-shelf* goal was only partially realized.

Although the prototype ITS is PC-based, it is not a toy system. Its task domain has more real-world elements than many larger ITSs running on much more expensive equipment. However, the subject matter is modest in its complexity, compared to say, radar training. Memory limitations will be severe as additional memory is added. It would be feasible to use the extended memory feature of the 80286 to circumvent this problem; however, since the Army-standard EIDS configuration is limited to 512K, moving to more complex topics could be even more difficult. To some extent, it should be possible to trade space against time, using overlays, at the expense of unfortunate programming complexity. The feasibility of this approach would be lessened to the degree that the topic chosen requires a large number of *simultaneously resident* knowledge chunks.

Figure 9: Sequence of Screens for Sample Session, Part N


You are in a combat situation and have just inserted a new magazine. The selector lever is on AUTO. The rifle fires once, then stops firing.



Use the mouse to select an action or a part of the rifle to concentrate on.

<u>Benefits Due to the Expert Module</u>. Many benefits or potential benefits derive from the availability of domain expertise within the training environment. The most apparent of these are described in the following paragraphs.

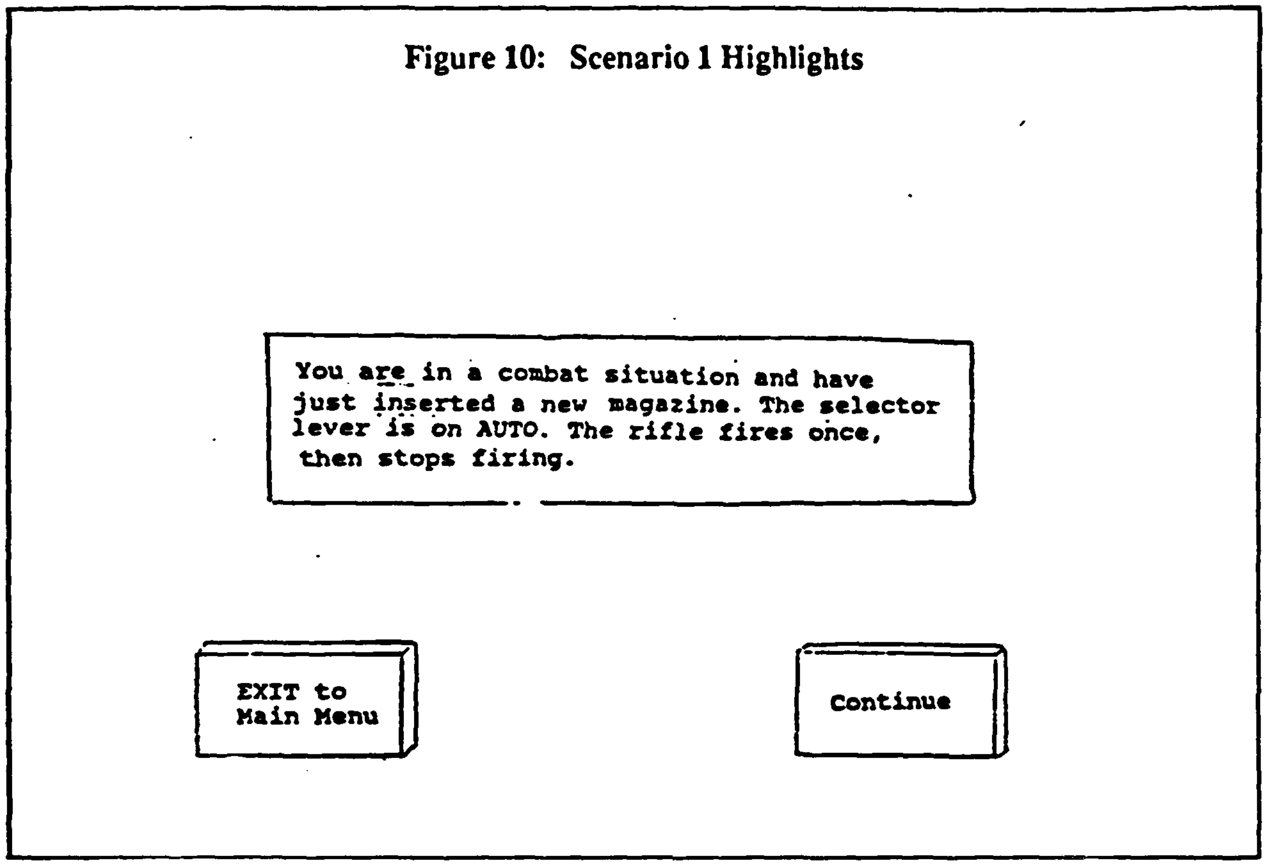
Hints and feedback are not "canned" or stored with the specific scenario, but instead represent advice that might be given by an actual expert placed in the student's current predicament.

By taking advantage of the *specific hint* capability at each step, the student can actually operate the system as if it were a stand-alone job aid; indeed, the *specific hint* feature supports a spectrum of usage from training to cooperative problem-solving to job aiding.

Depending on the degree of thought and care invested in structuring the knowledge base and associated explanatory information, students should perceive a level of capability or insight that is not apparent in conventional instructional systems.

The division of labor between domain expertise and presentation enables lesson authors to concentrate on the optimal presentation formats for instructional material, such as the appropriate use of high quality graphics, motion video, and sound. Answer judging, remedial sequencing, and similar decision-making, which can distract from the presentation and other aspects of the design of the training system, can be left to the expert module.

Figure 10: Scenario 1 Highlights

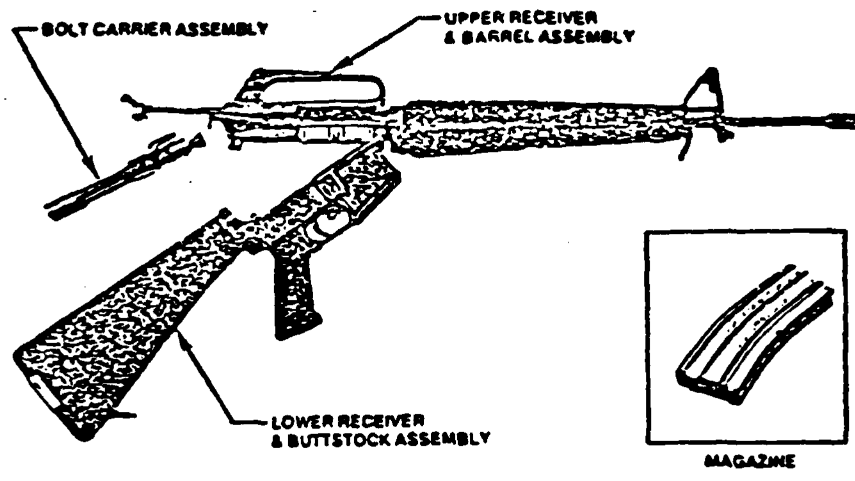

You are in a combat situation and have just inserted a new magazine. The selector lever is on AUTO. The rifle fires once, then stops firing.

It is far easier to add new scenarios than would be the case with a traditional courseware lesson. This extensibility results from the elimination of the requirement for the lesson author to consider every possible student outcome and prepare suitable responses and branching. It suffices to ensure that the expert module correctly solves the new scenario(s); it *is* necessary to ensure that the expert module can recover from student-chosen blind alleys on the new scenario. If the new scenario involves the same context and presenting symptom as an existing scenario, it may not even be necessary to prepare new video or graphics sequences; modifying the database to reflect the new underlying fault may be all that is required.

The potential exists for truly individualized tutoring, based on a fine-grained model of the student's knowledge and skills. It is, of course, possible to design ITS systems that use student models in the absence of an articulate (or human-like) expert system; the *Issues and Examples* approach of the *How the West Was Won* ITS system (Burton and Brown, 1982) illustrates one such system. However, it is far more convenient to model the student as an incomplete or partially incorrect version of an articulate expert. The current prototype was developed to support a student modelling system based on variations at the level of clauses in domain rules. Discussion of the planned student modelling capability appears below.

The potential exists for creating new scenarios dynamically, either at the request of the student, or by the training system itself applying standard variations to existing scenarios. For example, the student could conduct a "what if" analysis to better understand the expert's approach, by inserting alternate or additional faults that might be manifested by the same

Figure 11: Scenario 2 Highlights


BOLT CARRIER ASSEMBLY
UPPER RECEIVER & BARREL ASSEMBLY
LOWER RECEIVER & BUTTSTOCK ASSEMBLY
MAGAZINE

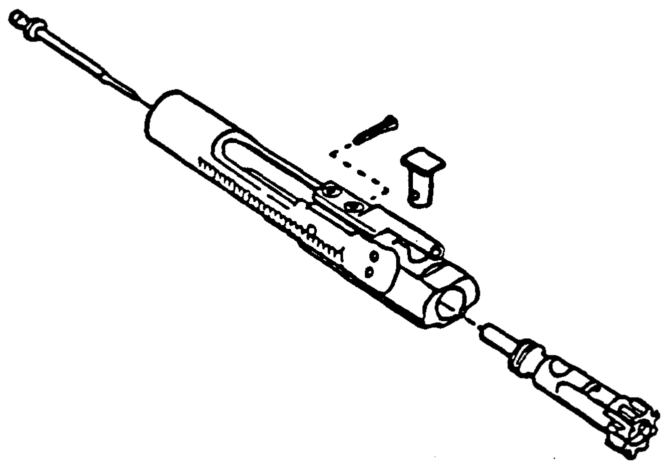
presenting symptom(s). Moreover, the training system could be made to generate variations on existing scenarios in order to exercise aspects of the knowledge (*eg*., certain rules) for which the student has completed the standard courseware lessons without demonstrating mastery; this assumes a well-elaborated student modelling module. The key to ensuring success for these types of capabilities would be ensuring that the resulting scenario variants could not fall out of the scope of coverage of the expert system.

<u>Benefits Due to the SAT/IVD Module</u>. What benefits or potential benefits, then, derive from the availability of the interactive videodisc and associated CBT authoring environment? A number of such benefits are readily apparent.

First and foremost, the instructional impact of the expert system is ***dramatically enhanced*** by the realism of the videodisc and the multi-media user interface. Too many current-generation ITS systems rely on little more than a ***glass teletype*** style of interaction, focusing on knowledge engineering at the expense of human engineering. Such systems, though impressive in laboratory demonstrations because of the underlying reasoning capabilities, can be ineffective in field testing with students who would prefer to watch television than exercise often-weak reading skills.

The lowest levels of input analysis, such as collection and timing of key presses and pointing device input, are handled by modular device drivers localized to the CBT package. The expert rules need not be concerned with whether the student indicated an affirmative response,

Figure 12: Scenario 3 Highlights



for example, by typing, eg., *Y* versus *y* versus *YES*, or by touching the *YES* box on a touch-sensitive screen, or by clicking the left mouse button while the cursor was over the *YES* region. This level of detail can be suppressed by the lesson author, and the abstract affirmative response signalled to the expert system.

Impossible or meaningless operations can be caught at the level of the authoring system and suppressed. This allows the expert system and the modelling modules to focus on the more interesting types of student errors, such as legal and meaningful but inappropriate steps for a given problem-solving context. The decision as to what level of granularity is appropriate for student modelling is a cooperative design decision to be made by the AI/ITS specialist and the CBT lesson author.

Too often in ITS work, a lack of awareness of relevant SAT literature can limit the instructional value of an otherwise powerful system. This results from an unfortunate lack of communication between the AI/ITS and SAT communities. Depending on the degree of thought and care invested in the instructional design, the pedagogical value of the expert system can therefore be significantly improved. Realizing this potential benefit depends upon the cooperation of development personnel in bringing their complementary backgrounds into harmony.

The division of labor between domain expertise and presentation enables the AI/ITS developers to concentrate on the optimal structuring and use of the domain knowledge. Screen layouts, presentation formats, selection of appropriate video sequences, and similar matters, which can distract from the knowledge engineering task, can be left to the training systems designers and lesson authors, who are often more attuned to such considerations.

Areas of Incompleteness. Schedule and budget constraints led to several areas of incompleteness in the current prototype; these are reviewed in the following paragraphs. It is important to distinguish these correctable flaws from *inherent limitations*, due to shortcomings of this particular approach or the state-of-the-art, which are examined later.

Since the focus of the work was on those aspects where an expert system could have a synergistic impact on training, other areas, where a conventional CBT approach might suffice, received less attention. In particular, only *stub* modules were developed for presenting introductory material about M16 riflery. The purpose of developing these stubs was to emphasize that the need for such lessons in a more complete system was in fact recognized, and to illustrate how such material might be integrated into a future release.

The current M16A1 expert system, involving about 250 rules, does not capture all relevant rifle knowledge. By design, it does not perform at depot level; it might also fail on some examples within its stated scope of rifle malfunctions reparable by field personnel. More importantly, because of memory limitations, the training system *per se* includes only the most essential subset of the expert rules.[7] Although the expert system correctly solves the four scenarios, its reasoning can seem somewhat superficial; this is partially attributable to the relative simplicity of the rifle troubleshooting task.

When providing feedback and hints, the current version of the prototype does not fully exploit the available domain knowledge. This is partly because only a subset of the expert rules are in use, but the primary reason is that not all rules incorporate the necessary explanatory information. For the expert system to be *fully articulate*, additional explanatory information would need to be added.

The current version does not actually demonstrate situations involving multiple, potentially interacting, underlying causes for malfunctions. This is not an inherent limitation, since the expert system, operating standalone, can solve such examples, and since an earlier version did successfully incorporate one such scenario. However, a lack of suitable video material contributed to a decision to remove that scenario from the delivered prototype. It will be important for future work to ensure that the capability to handle these sorts of complexities is clearly established.

As discussed earlier, the potential exists to dynamically generate scenario variations, either at the student's or the tutorial module's request. However, the current version of the system does not support this feature. A related capability would be allowing the student to backtrack to a prior move and select an alternative solution path. This idea was suggested by several audiences

[7]It is possible to operate the full troubleshooting expert system, separately, as a job aid, in order to access the remainder of the knowledge base. Although there are minor differences between the standalone version and the version that is integrated with TENCORE+, these could be eliminated with a small amount of additional effort.

upon seeing the finished demonstration. Since implementing this feature would simply require restoring a state vector from the existing session history, it would be straightforward to add it to the current prototype. In the meantime, the current system *does* allow trainees to try the same scenario repeatedly, thereby exploring the effects of alternative choices.

Some otherwise-legal troubleshooting operations are disallowed by the current prototype due to a lack of suitable video or graphics. In particular, take-down of certain sub-assemblies is forbidden only because a suitable frame illustrating that part of the rifle was not available. The result is a smaller search space for the student to consider, but this flaw might also lead to student misconceptions as to Army doctrine in this area. Such side effects were minimized in the current prototype, by selecting scenarios for which the restricted operations would have been inappropriate anyway.

Student modelling and advanced tutorial strategy features play a key role in the ITS system concept. Such features were designed, and plans for their eventual incorporation influenced a variety of design decisions ranging from knowledge representation to tool selection. However, overcoming the technical challenges involved in integrating the authoring environment with the expert module consumed more project resources than had been anticipated, precluding the completion of these features. Further investigation in this area is a high priority goal for future work.

A scenario wrap-up feature was also planned but not implemented in the current version. Originally, it was proposed that this module be implemented in the Ada® programming language.[8] The intent was to demonstrate the feasibility of interfacing existing, off-the-shelf tools with newly-written Ada-based modules, for delivery within a PC-based training system. A PC-targetted, validated Ada compiler was selected and purchased for the project, but the wrap-up feature was not completed in time for integration into the delivered prototype. It is expected that the Ada-based wrap-up feature will be completed in follow-on efforts.

Range of Applicability and Inherent Limitations. There are a variety of potential paradigms for using ITS technology in computer-based training. The approach taken in the current system appears to have broad applicability, but there are also certain inherent limitations. For some training situations, a more conventional SAT approach might be suitable; for others, an alternative ITS paradigm might serve. The strengths and weaknesses of the overall approach, as distinguished from the areas of incompleteness of the current demonstration system, are considered in the next few paragraphs.

The single most frustrating and time-consuming aspect of the project was the 640K byte memory limitation imposed by the DOS operating system. Each of TENCORE+ and OPS5+ could easily fill the entire 640K; both benefit from memory extension boards, when available. To handle maintenance training for devices more complex than the M16A1 rifle, it would be highly desirable to use an extended memory operating system, such as either OS II, or AI Architects'

[8]*Ada* is a registered trademark of the U.S. Department of Defense, Ada Joint Program Office. (There is growing momentum within the U.S. Army to standardize on the use of Ada for implementing all new mission-critical software systems.)

protected mode extensions to DOS. In parallel with this investigation, C*T has been developing an experimental version of OPS5+ based on the second approach. Additional memory would inevitably be required for expert-based maintenance training for highly complex weapons systems, such as the avionics bay of a modern Army helicopter.

Unfortunately, the Army-standard EIDS configuration imposes an even-more-severe 512K byte limitation on random access memory. For moderately complex troubleshooting domains, alternative solutions compatible with the EIDS requirement are feasible. These involve:

- using highly compact, stripped-down versions of the original tools;
- partitioning the knowledge base into small, modular rule sets; *and*
- overlaying large segments of the system to the hard disk.

The need to rely upon such techniques does not preclude applying the approach to somewhat more complex devices, but it does imply performance degradation, increased development cost, and an as-yet-to-be-determined upper bound on device complexity. The impact of memory restrictions becomes more severe as additional features, such as student modelling, are added.

One potential limitation is that some training topics could be awkward to represent using a production rule formalism such as OPS5. If an object-oriented or frame-oriented knowledge representation is required, the capability represented by the current prototype would not be sufficient. Although it would probably be possible to extend the approach to handle more complex knowledge representations, considerable additional research would be required.

Tremendous savings in development effort were achieved by using large, off-the-shelf components such as an existing CBT authoring environment and an existing expert system tool. However, there are corresponding weaknesses in terms of the *communications bandwidth* between these two modules, which in turn affected other design decisions. Some aspects of the knowledge, such as which operations can be performed on which sub-assemblies, had to be encoded redundantly, so as to make it accessible within each component. More generally, restrictions had to be placed on each module which could have been avoided if interaction with the other were more convenient. Moreover, the current system requires developers to use two distinct programming languages, or else to work through specialists in the operation of each language component in the hybrid authoring environment. This additional complexity would tend to make it more difficult to achieve another objective, facilitating the authoring of instructional materials by SMEs.

The availability of an expert system for the rifle troubleshooting task is crucial to the training paradigm employed by the current prototype. Yet, there are many topics, such as programmer training, where the development of an effective ITS system is feasible, even though development of a corresponding ES for the task would be beyond the state-of-the-art. The approach taken here is most applicable to training domains for which a high-performance expert system is attainable with reasonable effort. Likewise, the approach is most suitable for training situations which can be characterized as simulations or problem-solving scenarios; it seems less relevant to topics that are inherently declarative or that emphasize rote memorization or motor skills. Thus, M16A1 *troubleshooting* was chosen, whereas M16A1 *marksmanship* was *not* included as a topic in the demonstration system.

Plans for Future Investigation

The next major functional unit to be added to the prototype system involves student modelling. Currently, the data to initialize the student model is collected, based on the questions posed to actual students by instructors at the start of the rifle qualification course. Detailed evidence regarding which rules have been mastered is generated by the expert module at each step, by virtue of resetting its state to the actual situation faced by the trainee. This evidence would be used to model the student as a perturbation from the expert rules.

Two levels of granularity are possible. At one level, the *chunking* consists of a checklist of rules mastered or not mastered. Goldstein (1982) refers to this approach as an "overlay model." When the student makes a move that agrees with the expert move for that situation, this is positive evidence that the student has mastered the corresponding expert rule(s). When the student makes an incorrect move, this is negative evidence with respect to the corresponding expert rule(s). Although there may be multiple rules leading to the same choice, this potentially combinatoric problem is minimized by resetting the expert after each move, and can be resolved by differential diagnosis using situations where one rule applies but another does not.

Problems can arise in the presence of hints. If the trainee takes the specific hint and then makes a correct choice, this choice should not be treated as positive evidence. If the trainee takes the general hint and then makes a correct choice, it should be treated as positive evidence, but it should receive less weight. Modelling problems can also arise when the student selects a correct choice after first exploring a garden path; this situation can probably be handled by assuming correct rules with incorrect priorities or conflict resolution strategies.

The next finer level of granularity involves localizing flaws in the student's rule set to particular clauses in the antecedent or consequent part of a given rule. The trainee may be missing a clause from the left-hand side of a rule, for example, or may have an unnecessary condition on the applicability of a rule. The conclusion portion of a rule may be overly specific, and so on. Part of the key to making this type of modelling feasible lies in restricting the rule structure for the rules representing domain expertise, such as factoring out disjunctive conditions into separate rules, a discipline which has been observed in the ES for the M16A1.

A two stage modelling scheme can prevent the more fine-grained approach from searching too large a space of possible student models. Working first on a coarse granularity model (at the level of entire rules) and then refining the model to specific flaws within the buggy rules is analogous to the use of an abstract problem space in AI planning research. Our strategy, therefore, is to implement a rule-based modeller using this two-tier design, during the Phase II effort. It is also our plan to explore the generation of alternative scenarios (using the same video and symptom, for example, but different underlying faults) for differential diagnosis during student modelling. The results of this student modelling can then be used to guide: scenario selection (or generation) for remediation; presentation of hints within a given scenario; and modification of feedback after a given step.

Another area for investigation involves the tutorial output. An intriguing direction would be to provide additional categories of hint and advice, under control of the student. Just as the student can currently request either a general hint or a specific hint, other classes of hints could be offered in the pull-down menu. For example, a hint could rule out certain possibilities, or suggest a class of similar faults, or refer to a previously solved, related scenario. Eventually it might be possible to use the session history and student model in automatically selecting a hint category for spontaneous presentation without an explicit student request.

As a direction for longer-term research, a more optimal approach might be to develop an integrated authoring system, with support for both the AI/ES/ITS aspects and the SAT/IVD aspects, as well as for multiple training paradigms. Ideally, such a system would be accessible to SMEs who are neither ITS nor SAT specialists. Before attempting to define the specifications for such a sophisticated, integrated tool, however, it would be prudent for the community to acquire much more experience with successful ITS implementations using *ad hoc* techniques and off-the-shelf tools, along the lines pursued in this study.

Conclusion

This project clearly demonsrates the feasibility of integrating an intelligent tutoring system with interactive videodisc courseware, using a systems approach to training, for delivery on low-cost, PC-class hardware, such as the Army's new EIDS videodisc standard. The primary obstacle to scaling up the approach involves limits on random access memory, which can be circumvented by several well-understood programming techniques, for instructional materials of low to moderate complexity.

Several prevalent themes from the ITS research literature are embodied in the prototype system. The role of the domain ES in the driving the instructional dialogue is intriguing and unique. Other key concepts such as student modelling, which are incomplete or absent in the prototype, are plainly within the grasp of planned follow-on efforts. The required hardware platform can be delivered in high volume in a practical and cost effective form. The associated authoring environment, based on an interface between shelf software products, can be made completely generic with a small amount of additional work. Such an interface offers the potential to provide a widely available testbed for a hybrid ITS/SAT/IVD authoring tool. The associated instructional paradigm, though not a panacea for all military training needs, is broadly applicable to range of training requirements characterizable as problem-solving scenarios, gaming environments, and simulations, such as the diagnosis of faults in complex weapons systems or equipment.

Plans for further investigation include generalizing the OPS5+/TENCORE+ interface and adapting it to operate on EIDS, extending the approach to more complex topics such as digital electronics troubleshooting, fully implementing the student modelling and tutorial modules, and commencing field test and evaluation.

REFERENCES

Alessi, Stephen M., and Stanley R. Trollip (1985). *CBI: Methods and Develpment.* Englewood Cliffs, NJ: Prentice Hall.

Burton, R.R., and J.S. Brown (1982). An Investigation of Computer Coaching for Informal Learning Activities. In Sleeman, D., and J.S. Brown (Eds.), *Intelligent Tutoring Systems.* New York: Academic Press.

Brown, J.S., R.R. Burton, and J. de Kleer (1982). Pedagogical, Natural Language, and Knowledge Engineering Techniques in SOPHIE I, II, AND III. In Sleeman, D., and J.S. Brown (Eds.), *Intelligent Tutoring Systems.* New York: Academic Press.

Carbonell, J.R. (1970). AI in CAI: An Artificial Intelligence Approach to Computer Assisted Instruction. *IEEE Transactions on Man-Machine Systems MMS-11* (4).

De Bruin, J., Dan L. Massey, and Bruce Roberts (1988). A Training System for System Maintenance. In Psotka, J., L.D. Massey and S.A. Mutter (Eds.), *Intelligent Tutoring Systems: Lessons Learned.* Hillsdale, NJ: Lawrence Earlbaum Associates.

Fletcher, J.D. (1985). Intelligent Instructional Systems in Training. In Andriole, S.J. (Ed.), *Applications in Artificial Intelligence.* Princeton, NJ: Petrocelli Books.

Goldstein, I.P. (1982). The Genetic Graph: A Representation for the Evolution of Procedural Knowledge. In Sleeman, D., and J.S. Brown (Eds.), *Intelligent Tutoring Systems.* New York: Academic Press.

M16A1 Rifle: Operation and Maintenance Instructions (1980). Colt Manual CM101. Hartford, CT: Colt Firearms.

McDermott, J. (1982). R1: A Rule-based Configurer of Computer Systems. *Artificial Intelligence, 19* (1).

Organizational, Direct Support, and General Support Maintenance Manual (Including Repair Parts and Special Tools List): Rifle, 5.56-MM, M16 (1005-00-856-6885), Rifle, 5.56-MM, M16A1 (1005-00-073-9421) (1983). Technical Manual TM 9-1005-249-24&P, Washington, DC: Department of the Army.

Psotka, J., L.D. Massey and S.A. Mutter (eds.) (1988). *Intelligent Tutoring Systems: Lessons Learned.* Hillsdale, NJ: Lawrence Earlbaum Associates.

Sleeman, Derek, and John Seely Brown (1982). *Intelligent Tutoring Systems.* New York: Academic Press.

The M16A1 Rifle: Operation and Preventive Maintenance (1969). D.A. Pamphlet 750-30. Washington, DC: Department of the Army, U.S. Government Printing Office.

Waterman, Donald A. (1986). *A Guide to Expert Systems.* Reading, MA: Addison-Wesley.

Woolf, B.P. (1984). *Context-dependent Planning in a Machine Tutor* (Doctoral Dissertation). Amherst, MA: University of Massachusetts, Department of Computer and Information Science.

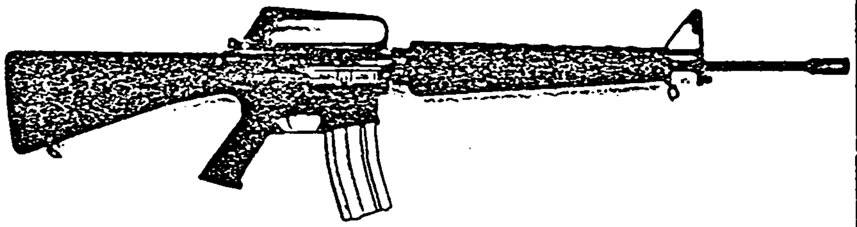
APPENDIX A
SAMPLE STORYBOARDS

REDIFFUSION
Simulation

COURSE:	M16A1 Expert Maint. & Troubleshooting	PAGE#	1
LESSON:	Scenario Two	PREPARED BY:	M. Wild
DATE:		APPROVED BY:	

M16A1 Expert Maintenance and Troubleshooting
Scenario Two

Combat -- Rifle Ceases Firing



EXIT to Main Menu

Continue

NEXT	
BACK	
HELP	
LAB	
INDEX	
DATA	

OTHER KEYS AND TOUCH AREAS
EXIT - Main Menu
Continue - p. 2

KEYBOARD	
TOUCH	
MOUSE	✓

VIDEODISC:
Motion
Still
Frames ____ to ____
Disc Segment
Audio Channel

COMPUTER:
Overlay
Graphic Files:

PROGRAMMER INSTRUCTIONS:

CONTINUED ON REVERSE

REDIFFUSION

COURSE:	M16A1 Expert Maint. & Troubleshooting	PAGE#	2
LESSON:	Scenario Two	PREPARED BY:	M. Wild
DATE:		APPROVED BY:	

You are in a combat situation and have just inserted a new magazine. The selector lever is on AUTO. The rifle fires once, then stops firing.

EXIT to Main Menu

Continue

NEXT	
BACK	
HELP	
LAB	
INDEX	
DATA	

OTHER KEYS AND TOUCH AREAS:
EXIT - Main Menu
Continue - A.3

KEYBOARD	
TOUCH	
MOUSE	✓

VIDEODISC:

Motion	✓
Still	✓
Frames	1500 to 1559
Disc Segment	
Audio Channel	

COMPUTER:

Overlay	✓
Graphic Files:	text

PROGRAMMER INSTRUCTIONS:

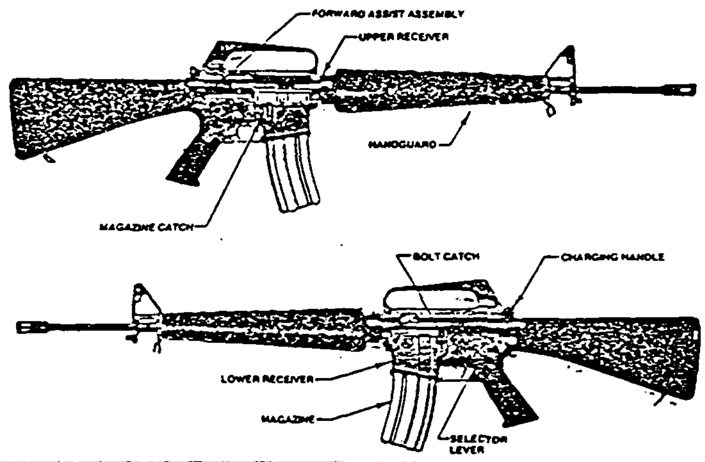
1. Play video and stop on last frame.
2. Do overlay text.

CONTINUED ON REVERSE

REDIFFUSION
Simulation

COURSE: M16A1 Expert Maint. & Troubleshooting PAGE# 3
LESSON: Scenario Two PREPARED BY: M. Wild
DATE: APPROVED BY:

HINTS REPLAY DISASSEM INSPECT REPAIRS TESTS



QUIT

Use the mouse to select an action or a part of the rifle to concentrate on.

NEXT
BACK
HELP
LAB
INDEX
DATA
OTHER KEYS AND TOUCH AREAS:

KEYBOARD
TOUCH
MOUSE

VIDEODISC:
Motion
Still
Frames to
Disc Segment
Audio Channel

COMPUTER:
Overlay
Graphic Files:

PROGRAMMER INSTRUCTIONS:

1. User can select component or an action in either order.
2. Highlight box for component selected.

CONTINUED ON REVERSE

APPENDIX B

SAMPLE TENCORE+ CODE

```
*
define   local
left     ,2
right    ,2
return   ,2
*
part     ,1       $$ which rifle part is selected
oldpart  ,1       $$ the last selected part
area     ,1       $$ which menu area has been chosen
xact     ,1       $$ repair activity
tact     ,1       $$ test action
hact     ,1       $$ hint action
icond    ,1       $$ condition of the inspected part
judge    ,1       $$ judging flag
cflag    ,1       $$ action return flag
define   end
*
initial  startup $$ for setup
QUIT     m16intro,index
display  2
screen   online
options  standard
video    init
video    image,off
do       m16lib,getdata  $$ restore data for user
*
if       scenario= 0
.        jump    m16intro,init
endif
*
do       m16lib,palset   $$ set up palette
do       m16lib,actable  $$ set up action table
*
*calc     rstatus ⇐ 1
calcc    rstatus<1 $or$ rstatus >5,rstatus ⇐ 1,,
do       rstatus $$ set up rifle condition / display
*
do       setdisp $$ set up display
*
calc     rpart ⇐ ract ⇐ 0        $$ init. actions
zero     oldx,8   $$ zero old box values
calc     oldpart ⇐ -1    $$ "no part"
*
lup
*
zero     part
zero     area
zero     return
enable   pointer,stream  $$ turn on mouse
*
*        Zero submenu items.
*
zero     submenu(1),240
loop
.        lloop
.        nextkey clear
.        pause   .5,keys=pointer
reloop   zkey=timeup
```

```
.       do       quit(;cflag) $$ check for QUIT box hit
.       branch   cflag,lsend,x
*
.       do       reassem(;cflag) $$ check for reassemble
.       branch   cflag,lout,x
*
.       if       (area1 $and$ left=down) $$ hint
.       .        calc     area ⇐ 1
.       .        pack     submenu(1),,Concept hint
.       .        pack     submenu(2),,Hint for next step
.       .        do       m16mouse,submenu(2,2 ; hact)
.       .        do       hint(hact)
.       .        branch   lloop
*
.       elseif   (area2 $and$ left=down) $$ replay
.       .        calc     area ⇐ 2
.       .        jump     replay
*
*       Replay this scenario information
*
.       elseif   (area3 $and$ left=down) $$ disassemble
.       .        calc     ract ⇐ 1          $$ first action
.       .        calc     area ⇐ 3
.       .        do       rpart=0;pickpart;x
.       elseif   (area4 $and$ left=down) $$ inspect
.       .        calc     ract ⇐ 2          $$ 2d action
.       .        calc     area ⇐ 4
.       .        do       rpart=0;pickpart;x
*
.       elseif   (area5 $and$ left=down) $$ correct
.       .        calc     area ⇐ 5
*
.       .        pack     submenu(1),,Send to Armorer
.       .        pack     submenu(2),,Lubricate with LSA--Heavy
.       .        pack     submenu(3),,Lubricate with LSA--Light
.       .        pack     submenu(4),,Replace with New Part
.       .        pack     submenu(5),,Clean
.       .        pack     submenu(6),,Adjust / Align
.       .        pack     submenu(7),,SPORTS sequence
.       .        pack     submenu(8),,Remove
.       .        do       m16mouse,submenu(5,8 ; xact)
.       .        calcc    xact>0,ract ⇐ 2+xact,,
.       .        if       ract > 0 $and$ rpart>0
.       .        .        branch  lsend
.       .        elseif   xact > 1
.       .        .        do      pickpart
.       .        endif
*
.       elseif   (area6 $and$ left=down) $$ test
.       .        calc     area ⇐ 6
.       .        pack     submenu(1),,Fire on Semi Auto
.       .        pack     submenu(2),,Fire on Full Auto
.       .        pack     submenu(3),,Chamber a Round
.       .        pack     submenu(4),,Dry Fire
.       .        pack     submenu(5),,Eject Round
.       .        pack     submenu(6),,Remove Cartridge
.       .       do      m16mouse,submenu(6,7 ; tact)
.       .       calcc   tact>0,ract <= 10+tact,,
.       .       if      ract > 0 $and$ rpart>0
.       .       .       branch  lsend
.       .       elseif  tact > 0
.       .       .       do      pickpart
.       .       endif
*
.       endif
outloop rpart>0 $and$ ract>0
endloop
lsend
do      decision(ract,rpart; judge)     $$ judge the act
branch  judge=NO,lup,x
disable pointer
*
lout
*
do      showact         $$ show part/action pair
pause   3       $$ give 'em time to read it
*
*  Now perform the requested action.
*
do      perform
do      m16lib,savedata $$ output results to disk
if      zauthsys=-1     $$ if authoring
.       press   SYS     $$ return to editor
else
.       pause   2       $$ wait...
.       exitsys         $$ return to OPS5+
endif
```

APPENDIX C

SAMPLE OPS5+ CODE

```
;;; no_extract malfunction rules

(p no_extract_1_q
   (m_no_extract_y)
   (get_new_info)
   -(answer ^qname t_no_extract_1 )
-->
   (make  test_or_inspect  ^name t_no_extract_1
      ^part magazine ^action inspect
      ^status cleared
      ^text "Is the magazine and ammo dirty or corroded?
      ^hint "Be sure and check other items that are place
   (make do_test t_no_extract_1)
#|   (make test_or_inspect ^name t_no_extract_1a
      ^part rifle_assembled_with_magazine ^action sports
      ^status assembled
      ^text "Has the SPORTS procedure been executed?"
      ^hint "Want to play a game?" )
   (make do_test t_no_extract_1a)
|#
   (make takedown_goal ^name clear_rifle ^hint |Make the
      ^text "Need to clear rifle to make safe before cont
   )

(p no_extract_1_y
   (m_no_extract_y)
   (get_new_info)
   (answer ^qname t_no_extract_1 ^result y)
-->
   (make correct_action  ^name c_clean_ammo_mag
      ^part ammunition ^action clean
      ^text "Remove ammunition and clean the mag."
      ^hint "Clean support items."
      ^status cleared )
   (make give_correct ^name c_clean_ammo_mag)
   )
```

Appendix C, Sample OPS5+ Code, Continued

```
(p no_extract_2q
   (m_no_extract_y)
   (get_new_info)
   (answer ^qname t_no_extract_1 ^result n)
  -(answer ^qname t_no_extract_2 )
-->
   (make  test_or_inspect  ^name t_no_extract_2
      ^part chamber ^action inspect
      ^status cleared
      ^text "Is there carbon and dirt build_up in chamber
      ^hint "Be sure and check places you don't normally
   (make do_test t_no_extract_2)
   )

(p no_extract_2y
   (m_no_extract_y)
   (get_new_info)
   (answer ^qname t_no_extract_2 ^result y)
-->
   (make correct_action  ^name c_no_extract_2
      ^part chamber ^action clean
      ^text "clean chamber."
      ^status cleared
      ^hint "Clean places you don't normally see.")
   (make give_correct ^name c_no_extract_2)
   )

(p no_extract_3er_q
   (m_no_extract_y)
   (get_new_info)
   (answer ^qname t_no_extract_2 ^result n)
   -(answer ^qname t_no_extract_3er )
-->
   (make  test_or_inspect  ^name t_no_extract_3er
      ^part extractor_recess ^action inspect
      ^status bolt_carrier_disassembled
      ^text "Carbon and dirt build_up in extractor recess
      ^hint "Check for dirty bolt parts." )
   (make do_test t_no_extract_3er)
   (make takedown_goal ^name diss_bolt ^hint |Look at ext
      ^text "Have to disassemble to bolt assembly before
   )
```

Appendix C, Sample OPS5+ Code, Continued

```
(p no_extract_3er_y
   (m_no_extract_y)
   (get_new_info)
   (answer ^qname t_no_extract_3er ^result y)
-->
   (make correct_action  ^name c_diss_and_clean
      ^part extractor_assembly ^action clean
      ^text "diss and clean."
      ^status bolt_carrier_disassembled
      ^hint "Better make it look better" )
   (make give_correct ^name c_diss_and_clean)
   )

(p no_extract_3el_q
   (m_no_extract_y)
   (get_new_info)
   (answer ^qname t_no_extract_3er ^result n)
   -(answer ^qname t_no_extract_3el )
-->
   (make  test_or_inspect  ^name t_no_extract_3el
      ^part extractor_assembly ^action inspect
      ^status bolt_carrier_disassembled
      ^text "Carbon and dirt build_up in extractor lip?
      ^hint "Is the lip okay?" )
   (make do_test t_no_extract_3el)
   )

(p no_extract_3el_y
   (m_no_extract_y)
   (get_new_info)
   (answer ^qname t_no_extract_3el ^result y)
-->
   (make correct_action  ^name c_diss_and_clean
      ^part extractor_assembly ^action clean
      ^text "diss and clean."
      ^status bolt_carrier_disassembled
      ^hint "Extract a clean one")
   (make give_correct ^name c_diss_and_clean)
   )
```

Appendix C, Sample OPS5+ Code, Continued

```
(p no_extract_4e_q
   (m_no_extract_y)
   (get_new_info)
   (answer ^qname t_no_extract_3el ^result n)
   -(answer ^qname t_no_extract_4e )
-->
   (make  test_or_inspect  ^name t_no_extract_4e
      ^part extractor_assembly ^action inspect
      ^status bolt_carrier_disassembled
      ^text "Is the extractor defective?"
      ^hint "Check for small defective items.")
   (make do_test t_no_extract_4e)
   )

(p no_extract_4e_y
   (m_no_extract_y)
   (get_new_info)
   (answer ^qname t_no_extract_4e ^result y)
-->
   (make correct_action  ^name c_replace
      ^part extractor_assembly ^action new_part
```

```
        ^text "Replace."
        ^status bolt_carrier_disassembled
        ^hint "May have to replace a certain assembly")
    (make give_correct ^name c_replace)
    )


(p no_extract_no_answer
    (m_no_extract_y)
    (get_new_info)
    (answer ^qname t_no_extract_4e ^result n)
-->
    (make re_test_rifle_2)
    )
```

APPENDIX D

KEY INTERFACES -- THEORY OF OPERATION

```
typedef unsigned char byte;

typedef struct tenops_dat
{

        byte user_data[256];     /* Name, Rank etc. Not used in program */
/* INFO WRITTEN BY TENCORE == 512 bytes */
        byte scen_count;         /* Scenario count
                                 ** 1 - Practice Range. No Fire
                                 ** 2 - Combat. Ceases Firing
                                 ** 3 - Functional Check. Fires on Safe
                                 ** 4 - Practice Range. Ceases Firing.
                                 */
        byte rifle_stat;         /* Rifle Status
                                 ** 1 - assembled
                                 ** 2 -                  mag assmbld
                                 ** 3 -                   "     "
                                 ** 4 -                   "     "
                                 ** 5 -                   "  dssmbld
                                 ** 6 -                   "     "
                                 ** 7 -                   "     "
                                 */

        byte rifle_part;         /* Rifle Part Chosen */
        byte rifle_act;          /* Rifle Action Chosen */
        byte step_count;         /* Number of Steps (part/action) made */
        byte FillTo32Bytes[27];  /* Filler bytes */
        byte actions[80][6];     /* Table of Student Actions
                                 ** 1 - Chosen Part
                                 ** 2 - Chosen Action
                                 ** 3 - 0 if no hint given
                                 ** 4 - Expected Part
                                 ** 5 - Expected Action
                                 ** 6 - Rifle Status
                                 */

/* INFO WRITTEN BY OPS5+ == 512 bytes */
        byte soft_hint[64];      /* General HINT on what to accomplish */
        byte part_hint;          /* Next Step Expected Part */
        byte act_hint;           /* Next Step Expected Action */
        byte feedback;           /* Feedback Type
                                 ** 0 - None
                                 ** 1 - Scenario Complete
                                 ** 2 - OK Proceed (getting warmer)
                                 ** 3 - Neutral Action (not good not bad )
                                 ** 4 - Negative NonFatal (getting colder )
```

```
                                  ** 5 - Fatal Error, Halt
                                  ** 6 - NonFatal, N/A, Undefined Previous
                                         Action.
                                  */
        byte scen_status;         /* Scenario Status
                                  ** 0 - Not Started
                                  ** 1 - Not Completed
                                  ** 2 - Finished
                                  */
        byte fill_to_512[444]; /* filler bytes */

} TENOPS_DAT, *TENOPS_DATP;
```